CLAIMS

What is claimed is:

1. A method comprising:

processing network traffic using a first program, the first program containing a first interface instance having a first behavior;

detecting a first condition;

generating a second program, the second program containing a second interface instance having a second behavior, the generation of the second program including selecting the second interface instance from a plurality of interface instances for inclusion in the second program; and

processing network traffic using the second program.

2. The method of claim 1, in which the second interface instance is inlined into the second program, such that it is reachable without executing a jump or branch instruction.

3. The method of claim 1, in which generating the second program comprises:

interpreting a switch statement in a master program to locate the second interface instance; and

removing one or more interface instances other than the second interface instance from the master program.

4. The method of claim 3, in which the one or more interface instances other than the second interface instance includes the first interface instance.

5. The method of claim 3, in which the generation of the second program is performed by a linker.

6. The method of claim 1, further comprising replacing the first program with the second program in at least one microengine, wherein the processing of network traffic using the first and second programs is performed by said at least one microengine.

7. The method of claim 1, in which the first condition comprises a change in network traffic.

8. The method of claim 1, further comprising:

detecting a second condition;

generating a third program, the third program containing a third interface instance having a third behavior, the generation of the third program including selecting the third interface instance from a plurality of interface instances for inclusion in the third program; and

processing network traffic using the third program.

9. The method of claim 8, in which the third program includes the second interface instance, the second interface instance corresponding to a different interface from the third interface instance.

10. The method of claim 1, in which generating the second program comprises replacing a subroutine call in a copy of a master program with the second interface instance.

11. The method of claim 1, in which generating the second program comprises removing code from a third program.

12. The method of claim 11, in which the second program is smaller than the third program.

13. The method of claim 1, in which the first program and the second program comprise different versions of the same program.

14. The method of claim 1, in which the first program and the second program are written in an instruction set of a microengine that performs the processing of network traffic using the first and second programs.

15. A computer program package embodied on a computer readable medium, the computer program package including instructions that, when executed by a processor, cause the processor to perform actions comprising:

obtaining an identification of a selected implementation of a first interface;

obtaining a first code image containing the selected implementation of the first interface and one or more other implementations of the first interface;

generating a second code image by removing the one or more other implementations of the first interface from the first code image.

16. The computer program package of claim 15, in which the selected implementation of the first interface and the one or more other implementations of the first interface are located in a switch statement in the first code image.

17. The computer program package of claim 15, in which the computer readable medium comprises a memory unit associated with a network processor, and in which the processor comprises a core processor of said network processor.

18. A system comprising:

a network processor comprising:

a processing core;

one or more microengines; and

a memory unit, the memory unit including code that, when executed by the processing core, is operable to cause the network processor to perform actions comprising:

detecting a first condition;

identifying a first instance of a first interface suitable for handling the first condition;

selecting the first instance of the first interface from a plurality of instances of the first interface;

generating a code image that includes the first instance of the first interface; and

loading the code image into one or more of the microengines for execution.

19. The system of claim 18, in which the plurality of instances of the first interface are arranged within a switch statement in a master code image stored in memory accessible by said processing core.

20. A system as in claim 18, in which the memory unit further includes a linker, the linker being operable to interpret a switch statement and to remove unselected instances of an interface from the switch statement.

21. A system as in claim 18, in which the memory unit further includes an instance resolver, the instance resolver including the code for detecting a first condition and identifying a first instance of a first interface suitable for handling the first condition.

22. The system of claim 18, further comprising:

a computer system to enable development of software for use on the network processor, the computer system including:

a compiler operable to compile a source code program into an object code program, the compiler being operable to inline said plurality of instances of the first interface into the object code program.

23. A method for performing dynamic resource adaptation, the method comprising:

identifying a selected interface implementation;

removing one or more other interface implementations from a first code image to form a second code image that includes the selected interface implementation;

using the second code image to perform one or more network processing tasks.

24. The method of claim 23, in which the first code image contains a switch statement that includes the selected interface implementation and the one or more other interface implementations, the method further comprising:

removing the switch statement from the first code image.

25. The method of claim 23, in which removing of the one or more other interface implementations is performed by a linker.

26. A system comprising:

a switch fabric; and

one or more line cards comprising:

one or more physical layer components; and

one or more network processors, at least one of said network processors comprising:

a processing core;

one or more microengines; and

a memory unit, the memory unit including code that, when executed by the processing core, is operable to cause the network processor to perform actions comprising:

detecting a first condition;

identifying a first instance of a first interface suitable for handling the first condition;

selecting the first instance of the first interface from a plurality of instances of the first interface;

generating a code image that includes the first instance of the first interface; and

loading the code image into one or more of the microengines for execution.

27. A system as in claim 26, in which the memory unit further includes a linker, the linker being operable to interpret a switch statement and to remove unselected instances of an interface from the switch statement.

28. A system as in claim 26, in which the memory unit further includes an instance resolver, the instance resolver including the code for detecting a first condition and identifying a first instance of a first interface suitable for handling the first condition.